Proof obligation discharge using the PF transform

J.N. Oliveira

Dept. Informática,
Universidade do Minho
Braga, Portugal

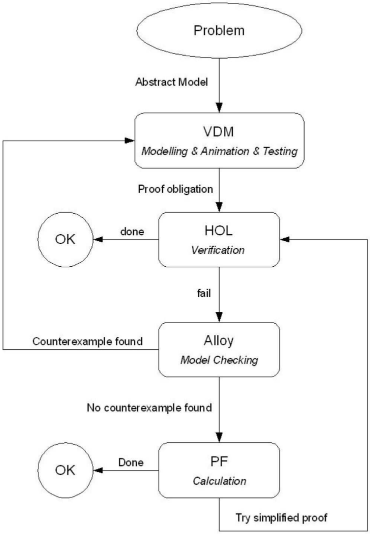
DI/UM, 2008

Summary

Learning outcomes:

- Discharging proof obligations via PF-transform. Pre/post conditions. Invariants.
- Extended static checking in the PF-style. PF-calculation of weakest pre-conditions for invariant maintenance.
- Three examples

Broad picture: a "all-in-one" strategy for PO discharge



Poof obligations in the PF-style

In general:

Input/output property preservation (functions)

Proof obligation

$$\langle \forall\, x \;:\; p\, x :\; q\,(f\; x)\rangle \tag{1}$$

stating that function f **ensures** *property* q *on its* **output** *every time property* p *holds on its* **input** *PF-transforms to*

$$f \cdot \Phi_p \;\subseteq\; \Phi_q \cdot f \quad \textit{cf. diagram} \quad \begin{tikzcd} A \arrow[d, "f"'] & A \arrow[l, "\Phi_p"'] \arrow[d, "f"] \\ B & B \arrow[l, "\Phi_q"] \end{tikzcd} \tag{2}$$

Predicates as "types"

We will write "type declaration"

$$\Phi_q \xleftarrow{f} \Phi_p \tag{3}$$

to mean (2).

Exercise 1: Show that (2) and

$$f \cdot \Phi_p \;\subseteq\; \Phi_q \cdot \top \tag{4}$$

are the same.

$\square$

Exercise 2: Prove the equivalence

$$\Phi_q \xleftarrow{id} \Phi_p \quad \equiv \quad q \Leftarrow p \tag{5}$$

$\square$

Exercises

Exercise 3: Infer from (3) and properties (59) to (61) the following ESC (*extended static checking*) properties:

$$\Phi_q \xleftarrow{f} \Phi_{p_1} \cup \Phi_{p_2} \quad \equiv \quad \Phi_q \xleftarrow{f} \Phi_{p_1} \ \wedge \ \Phi_q \xleftarrow{f} \Phi_{p_2} \tag{6}$$

$$\Phi_{q_1} \cdot \Phi_{q_2} \xleftarrow{f} \Phi_p \quad \equiv \quad \Phi_{q_1} \xleftarrow{f} \Phi_p \ \wedge \ \Phi_{q_2} \xleftarrow{f} \Phi_p \tag{7}$$

□

Exercise 4: Using (4) and the relational version of McCarthy's conditional combinator which follows,

$$c \rightarrow f, g \ = \ f \cdot \Phi_c \cup g \cdot \Phi_{\neg c} \tag{8}$$

infer the *conditional ESC* rule which follows:

$$\Phi_q \xleftarrow{c \rightarrow f, g} \Phi_p \quad \equiv \quad \Phi_q \xleftarrow{f} \Phi_p \cdot \Phi_c \ \wedge \ \Phi_q \xleftarrow{g} \Phi_p \cdot \Phi_{\neg c} \tag{9}$$

□

Relationship with Hoare Logic

Let us show that **Hoare triples** such as

$$\{p\}P\{q\} \tag{10}$$

are also instances of ESC proof obligations. First we spell out the meaning of (10):

$$\langle \forall\, s \;:\; p\ s : \; \langle \forall\, s' \;:\; s \xrightarrow{P} s' \;:\; q\ s' \rangle \rangle \tag{11}$$

Then (recording the meaning of program P as relation $[\![P]\!]$ on program states) we PF-transform (11) into

$$\Phi_p \subseteq [\![P]\!] \setminus (\Phi_q \cdot \top) \tag{12}$$

thanks to the introduction of relational (left) **division**,

$$b\ (R \setminus S)\ a \;\equiv\; \langle \forall\, c \;:\; c\ R\ b : \; c\ S\ a \rangle \tag{13}$$

Relationship with Hoare Logic

Thanks to "al-djabr" rule

$$R \cdot X \subseteq S \quad \equiv \quad X \subseteq R \setminus S \tag{14}$$

we obtain

$$[\![P]\!] \cdot \Phi_p \subseteq \Phi_q \cdot \top \tag{15}$$

equivalent to

$$[\![P]\!] \cdot \Phi_p \subseteq \Phi_q \cdot [\![P]\!]$$

which shares the same scheme as

$$f \cdot \Phi_p \ \subseteq \ \Phi_q \cdot f$$

earlier on.

Summary

In general, we will write "type declaration"

$$\Psi \xleftarrow{R} \Phi \tag{16}$$

to mean

$$R \cdot \Phi \subseteq \Psi \cdot R \tag{17}$$

In words:

- Notation (16) can be regarded as the **type assertion** that, if fed with values (or starting on states) "of type Φ" computation R yields results (moves to states) "of type Ψ" (if it terminates).
- So functional ESC POs and Hoare triples are one and the same device: a way to **type** computations, be them specified as (always terminating, deterministic) functions or encoded into (possibly non-terminating, non-deterministic) programs.

The invariant maintenance (IM) PO

Pointfree:

$$\Phi_{inv} \xleftarrow{R} \Phi_{inv} \tag{18}$$

that is,

$$R \cdot \Phi_{inv} \quad \subseteq \quad \Phi_{inv} \cdot R \tag{19}$$

Pointwise (functions):

$$\langle \forall\ a\ :\ inv\ a :\ inv(f\ a)\rangle \tag{20}$$

Pointwise (relations):

$$\langle \forall\ a\ :\ inv\ a :\ \langle \forall\ a'\ :\ a'\ R\ a :\ inv\ a'\rangle\rangle \tag{21}$$

Mid point: pre-conditioned functions

The most typical situation corresponds to R being a function restricted by some precondition:

- Let $R := f \cdot \Phi_{pre}$ in (18), where pre is a given precondition.
- Then (18) becomes

$$
\begin{array}{ll}
& \Phi_{inv} \xleftarrow{f \cdot \Phi_{pre}} \Phi_{inv} \\
\equiv & \qquad \{ \text{ definition } \} \\
& f \cdot \Phi_{pre} \cdot \Phi_{inv} \subseteq \Phi_{inv} \cdot \top \\
\equiv & \qquad \{ \text{ definition } \} \\
& \Phi_{inv} \xleftarrow{f} \Phi_{pre} \cdot \Phi_{inv} \\
\equiv & \qquad \{ \text{ going pointwise } \} \\
& \langle \forall\ a\ ::\ pre\ a \wedge inv\ a \Rightarrow inv(f\ a) \rangle
\end{array} \tag{22}
$$

Calculating Preconditions for IM

- Very often f and inv are given and pre is the "unknown": the idea is to find pre which is "enough" for (22) to hold.
- In fact, wherever f does not ensure maintenance of invariant inv, there is always a **pre-condition** pre which enforces this at the cost of *partializing* f: in the limit, pre is the *everywhere false* predicate.
- As a rule, the average programmer will become aware of such a pre-condition at runtime, in the **testing** phase.
- One can find it much earlier, at specification time, when trying to discharge the standard proof obligation (22).

PF-ESC instead of invent & verify

However,

- Bound to **invent** pre, we'll hope to have guessed the **weakest** such pre-condition. Otherwise, future use of f will be spuriously constrained.
- Can we be sure of having hit the **weakest** pre-condition?

Our approach (**PF-ESC**) will be as follows:

- We take the PF-transform of $inv(f\ a)$ in (22) — at data level — and attempt to rewrite it to a term involving $inv\ a$ and possibly "something else": the **calculated** pre-condition.
- This will be the weakest provided the calculation stays within equivalence steps (as shown in the next slides).

Weakest pre-conditions

- Let us strengthen (22) to equivalence

$$\langle \forall\, a \;::\; (pre\ a) \wedge (inv\ a) \equiv inv(f\ a) \rangle \tag{23}$$

which PF-transforms to equality

$$\Phi_{pre} \cdot \Phi_{inv} = \top \cdot \Phi_{inv} \cdot f \tag{24}$$

- Later on we will show that (24) ensures *pre* as the **weakest** (up to logical equivalence) pre-condition for *inv* to be preserved.
- Weakest = sufficient + necessary for $inv(f\ a)$ to hold.

Case study 1: PF-ESC at work

We want to calculate the WP for

$$add\ x\ l \;\;\triangleq\;\; x : l$$

to preserve the **no duplicates** invariant on finite lists.

- First step: PF-transform $X^{\star}$ to $\mathbb{N} \rightharpoonup X$ (**simple** relation telling which elements take which position in list).

  Then the **no duplicates** invariant on L is encoded as $\ker L \subseteq id$ (L is injective)

  Finally, $add\ x\ L$ PF-transforms to

$$\underline{x} \cdot \underline{1}^{\circ} \cup L \cdot succ^{\circ} \qquad (25)$$

  cf. back to points: $\{1 \mapsto x\} \cup \{i+1 \mapsto (L\ i) : i \leftarrow \delta\ L\}$.

Case study 1: PF-ESC at work

- Second step: we start from the right hand side $inv(add\ x\ L)$ of (23) and re-write it by successive equivalence steps until we reach:
  - condition $inv\ l$ ...
  - ... "plus something else" — the calculated weakest pre-condition.
- Since the PF-transformed proof has to do with injectivity of union of relations, the following fact

$$R \cup S \text{ is injective } \equiv \qquad\qquad R \text{ is injective} \wedge S \text{ is injective} \wedge R^{\circ} \cdot S \subseteq id \qquad (26)$$

(easy to prove) is likely to be of use.

Case study 1: PF-ESC at work

$add\ x\ L$ has no duplicates

$\equiv \quad \{$ cf. (25) etc $\}$

$\underline{x} \cdot \underline{1}^\circ \cup L \cdot succ^\circ$ is injective

$\equiv \quad \{$ (26) $\}$

$\underline{x} \cdot \underline{1}^\circ$ is injective $\wedge\, L \cdot succ^\circ$ is injective $\wedge\, (\underline{x} \cdot \underline{1}^\circ)^\circ \cdot L \cdot succ^\circ \subseteq id$

$\equiv \quad \{$ definition of injective (twice) ; "al-djabr" (59) $\}$

$\underline{1} \cdot \underline{x}^\circ \cdot \underline{x} \cdot \underline{1}^\circ \subseteq id \wedge succ \cdot L^\circ \cdot L \cdot succ^\circ \subseteq id \wedge \underline{x}^\circ \cdot L \subseteq \underline{1}^\circ \cdot succ$

$\equiv \quad \{$ "al-djabr" (59,60) as much as possible $\}$

$\underline{x}^\circ \cdot \underline{x} \subseteq \underline{1}^\circ \cdot \underline{1} \wedge L^\circ \cdot L \subseteq succ^\circ \cdot succ \wedge \underline{x}^\circ \cdot L \subseteq \underline{1}^\circ \cdot succ$

$\equiv \quad \{$ kernel of constant function is $\top$; *succ* is an injection $\}$

$\textsc{True} \wedge L^\circ \cdot L \subseteq id \wedge \underline{x}^\circ \cdot L \subseteq \underline{1}^\circ \cdot succ$

Case study 1: summary

We have thus calculated:

$$\textit{add}\ x\ L \text{ has no duplicates} \quad \equiv \quad \underbrace{L \text{ is injective}}_{\text{no duplicates in } L} \ \wedge\ \underbrace{\underline{x}^\circ \cdot L \subseteq \underline{1}^\circ \cdot \textit{succ}}_{\text{WP}}$$

PW-expansion of the calculated WP:

$$
\begin{array}{ll}
& \underline{x}^\circ \cdot L \subseteq \underline{1}^\circ \cdot \textit{succ} \\
\equiv & \quad \{ \text{ go pointwise: (48) twice } \} \\
& \langle \forall\ n\ ::\ x\ L\ n \Rightarrow 1 = 1 + n \rangle \\
\equiv & \quad \{ \ L \text{ models list } l \ \} \\
& \langle \forall\ n\ :\ n \in \text{inds}\, l :\ x = (l\ n) \Rightarrow 1 = 1 + n \rangle \\
\equiv & \quad \{ \ 1 = 1 + n \text{ always false } (n \in \mathbb{N}) \ \} \\
& \langle \forall\ n\ :\ n \in \text{inds}\, l :\ (l\ n) \neq x \rangle
\end{array}
$$

Case study 2: PF-ESC at work

From the mobile phone directory problem we select maintenance of the no duplicates invariant by function

$$store\ x \quad \triangleq \quad (take\ 10) \cdot (x\ :) \cdot filter(x \neq)$$

Remarks:

- It's sufficient to show that $(x\ :) \cdot filter(x \neq)$ preserves injectivity, since $take\ n\ L \ \subseteq\ L$ $(\forall n)$ and *smaller than injective is injective*
- Defined over PF-transformed lists, $filter$ becomes

$$filter(x \neq)L \quad \triangleq \quad (\neg\rho\, \underline{x}) \cdot L \tag{27}$$

where the negated range operator $(\neg\rho)$ satisfies property

$$\Phi \subseteq \neg\rho\, R \quad \equiv \quad \Phi \cdot R \subseteq \bot \tag{28}$$

Case study 2: PF-ESC at work

$$
\begin{array}{ll}
& x : (\mathit{filter}(x \neq) L) \text{ is injective} \\
\equiv & \quad \{ \text{ case study 1, (27) } \} \\
& (\neg \rho\, \underline{x}) \cdot L \text{ is injective} \wedge \underline{x}^{\circ} \cdot (\neg \rho\, \underline{x}) \cdot L \subseteq \underline{1}^{\circ} \cdot \mathit{succ} \\
\Leftarrow & \quad \{ \text{ smaller than injective is injective } \} \\
& L \text{ is injective} \wedge \underline{x}^{\circ} \cdot (\neg \rho\, \underline{x}) \cdot L \subseteq \underline{1}^{\circ} \cdot \mathit{succ} \\
\equiv & \quad \{ \text{ converses } \} \\
& L \text{ is injective} \wedge L^{\circ} \cdot (\neg \rho\, \underline{x}) \cdot \underline{x} \subseteq \mathit{succ}^{\circ} \cdot \underline{1} \\
\equiv & \quad \{ (\neg \rho\, \underline{x}) \cdot \underline{x} = \bot \text{ by left-cancellation of (28) } \} \\
& L \text{ is injective} \wedge L^{\circ} \cdot \bot \subseteq \mathit{succ}^{\circ} \cdot \underline{1} \\
\equiv & \quad \{ \text{ bottom is below anything } \} \\
& L \text{ is injective} \wedge \textsc{True}
\end{array}
$$

Case study 2: PF-ESC at work

Moral of this case study:

> *Although the implication in the second step of the reasoning could put weakness of calculated pre-condition at risk, we've calculated the weakest of all conditions anyway (TRUE).*

Exercise 5: Show that (28) stems from "al-djabr" rule

$$\Phi \subseteq \neg\delta\, R \quad \equiv \quad R \subseteq \bot/\Phi \tag{29}$$

among others.

□

Exercise 6: Prove (26).

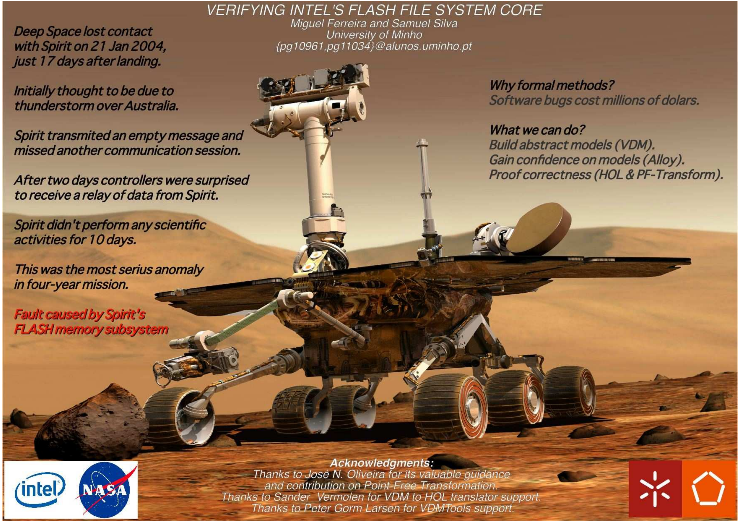
□

Case study 3: Verified File System

A real-life case study:

- **VSR** (Verified Software Repository) initiative
- **VFS** (Verified File System) on Flash Memory — challenge put forward by Rajeev Joshi and Gerard Holzmann (NASA JPL) [2]
- Two levels — POSIX level and (NAND) flash level
- Working document: **Intel ® Flash File System Core Reference Guide** (Oct. 2004) is POSIX aware.

Case study 3: Verified File System



Case study 3: Verified File System

The problem (sample):

File System API Reference

4.6 FS_DeleteFileDir

Deletes a single file/directory from the media

Syntax

```
FFS_Status    FS_DeleteFileDir (
        mOS_char      *full_path,
        UINT8         static_info_type );
```

Parameters

Parameter	Description
*full_path	(IN) This is the full path of the filename for the file or directory to be deleted.
static_info_type	(IN) This tells whether this function is called to delete a file or a directory.

Error Codes/Return Values

FFS_StatusSuccess	Success
FFS_StatusNotInitialized	Failure
FFS_StatusInvalidPath	Failure
FFS_StatusInvalidTarget	Failure
FFS_StatusFileStillOpen	Failure

Verified File System Project

Sample of model's data types (simplified):

$$System = \{table : OpenFileDescriptorTable, tar : Tar\}$$
$$\textbf{inv} \;\; sys \triangleq \langle \forall\, ofd \; : \; ofd \in rng\ (table\ sys) : \; path\ ofd \in dom\ tar\ sys \rangle$$

where

$$OpenFileDescriptorTable = FileHandler \rightharpoonup OpenFileDescriptor$$

$$Tar = Path \rightharpoonup File$$
$$\textbf{inv} \;\; tar \triangleq \langle \forall\, p \; : \; p \in dom\ tar : \; dirName(p) \in dom\ tar \land fileType(attributes(tar(dirName\ p))) = Directory \rangle$$

$$OpenFileDescriptor = \{path : Path, \ldots\}$$

Verified File System Project

(Sample) API function:

$$FS\_DeleteFileDir : Path \rightarrow System \rightarrow (System \times FFS\_Status)$$

$$FS\_DeleteFileDir\ p\ sys \triangleq$$

$$\quad if\ \ p \neq Root \wedge p \in dom\ (tar\ \ sys) \wedge \text{pre-}FS\_DeleteFileDir\_System\ p\ sys$$

$$\quad then\ (FS\_DeleteFileDir\_System\ p\ sys, FFS\_StatusSuccess)$$

$$\quad else\ (sys, FS\_DeleteFileDir\_Exception\ p\ sys)$$

where

$$FS\_DeleteFileDir\_System : Path \rightarrow System \rightarrow System$$

$$FS\_DeleteFileDir\_System\ p\ (h, t) \triangleq$$

$$\quad (h, FS\_DeleteFileDir\_Tar\ \{p\}\ t)$$

$$\mathbf{pre} \left\langle \begin{array}{c} \forall\ buffer \\ buffer \in rng\ \ h : \\ path\ buffer \neq p \wedge \text{pre-}FS\_DeleteFileDir\_Tar\ \ p\ \ t \end{array} \right\rangle$$

Verified File System Project

Sample API function (continued):

$$FS\_DeleteFileDir\_Tar : \mathcal{P}\,Path \rightarrow Tar \rightarrow Tar$$
$$FS\_DeleteFileDir\_Tar\ s\ t \triangleq\ tar \setminus s$$
$$\textbf{pre}\ \langle \forall\ p\ :\ p \in dom\ tar :\ dirName\ p \in s \Rightarrow p \in s \rangle;$$

where

$$dirName : Path \rightarrow Path$$
$$dirName\ p \triangleq\ if\ p = Root \vee len\ p = 1$$
$$then\ Root$$
$$else\ blast\ p$$

and so on. (**NB:** *blast* selects all but the last element of a list.)

Invariant structural synthesis (coreflexives)

- Real-size problems show **where complexity is**, namely the intricate structure involving nested datatype invariants.
- Need to calculate the associated coreflexives.
- Denoting by A_p the fact that datatype A is constrained by invariant p, we will write $\in_{A_p}$ to denote the associated coreflexive, calculated by induction on the structure of types:

$$\in_X = id \tag{30}$$

$$\in_{K_p} = \Phi_p \tag{31}$$

$$\in_{(A \times B)_p} = (\in_A \times \in_B) \cdot \Phi_p \tag{32}$$

$$\in_{(A+B)_{[p\ ,q]}} = \in_A \cdot \Phi_p + \in_B \cdot \Phi_q \tag{33}$$

$$\in_{(\mathsf{F}\,A)_p} = \mathsf{F}(\in_A) \cdot \Phi_p \tag{34}$$

Invariant structural synthesis (coreflexives)

Example:

$$
\begin{array}{ll}
& \in_{System} \\
= & \qquad \{ \text{ (32), for } ri \text{ (= "referential integrity") the top level inv. } \} \\
& (\in_{OpenFileDescriptorTable} \times \in_{Tar}) \cdot \Phi_{ri} \\
= & \qquad \{ \text{ } OpenFileDescriptorTable \text{ has no invariant } \} \\
& (id \times \in_{Tar}) \cdot \Phi_{ri} \\
= & \qquad \{ \text{ (31) for } pc \text{ (= "prefix closed") denoting } Tar\text{'s invariant } \} \\
& (id \times \Phi_{pc}) \cdot \Phi_{ri}
\end{array}
\tag{35}
$$

Facing complexity

Need to "find structure" in the specification text:

- *FS_DeleteFileDir p* has conditional "shape"

$$c \longrightarrow \langle f \cdot \Phi_p, \underline{k} \rangle, \langle id, g \rangle \qquad (36)$$

where

- c is the (main) if-then-else's condition
- f abbreviates *FS_DeleteFileDir_System p*
- p is the precondition of f
- k abbreviates *FFS_StatusSuccess*
- g abbreviates *FS_DeleteFileDir_Exception p*

What's the advantage of pattern (36)?

See the "divide and conquer" rules which follow:

Breaking complexity of POs

Further to (5), (7), (9):

- **Trivial**:

$$id \xleftarrow{R} \Phi \quad \equiv \quad \Phi \xleftarrow{R} \bot \quad \equiv \quad \Psi \xleftarrow{\bot} \Phi \quad \equiv \quad \textsc{True} \qquad (37)$$

- **Trading**:

$$\Upsilon \xleftarrow{R} \Phi \cdot \Psi \quad \equiv \quad \Upsilon \xleftarrow{R \cdot \Phi} \Psi \qquad (38)$$

- **Composition** (**Fusion**):

$$\Psi \xleftarrow{R \cdot S} \Phi \quad \Leftarrow \quad \Psi \xleftarrow{R} \Upsilon \wedge \Upsilon \xleftarrow{S} \Phi \qquad (39)$$

Breaking complexity of POs

- **Split by conjunction**:

$$\Psi_1 \cdot \Psi_2 \xleftarrow{R} \Phi \quad \equiv \quad \Psi_1 \xleftarrow{R} \Phi \;\wedge\; \Psi_2 \xleftarrow{R} \Phi \tag{40}$$

— generalizes (7)

- **Weakening/strengthening**:

$$\Psi \xleftarrow{R} \Phi \quad \Leftarrow \quad \Psi \supseteq \Theta \wedge \;\Theta \xleftarrow{R} \Upsilon \;\wedge \Upsilon \supseteq \Phi \tag{41}$$

- **Separation**:

$$\Upsilon \cdot \Theta \xleftarrow{R} \Phi \cdot \Psi \quad \Leftarrow \quad \Upsilon \xleftarrow{R} \Phi \;\wedge\; \Theta \xleftarrow{R} \Psi \tag{42}$$

— outcome of (41), (40)

Breaking complexity of POs

- **Splitting** (functions):

$$\Psi \times \Upsilon \xleftarrow{\langle f,g \rangle} \Phi \quad \equiv \quad \Psi \xleftarrow{f} \Phi \ \wedge \ \Upsilon \xleftarrow{g} \Phi \tag{43}$$

- **Splitting** (in general):

$$\Psi \times \Upsilon \xleftarrow{\langle R,S \rangle} \Phi \quad \equiv \quad \Psi \xleftarrow{R} \Phi \cdot \delta\, S \ \wedge \ \Upsilon \xleftarrow{S} \Phi \cdot \delta\, R \tag{44}$$

- **Product**:

$$\Phi' \times \Psi' \xleftarrow{R \times S} \Phi \times \Psi \quad \equiv \quad \Phi' \xleftarrow{R} \Phi \ \wedge \ \Psi' \xleftarrow{S} \Psi \tag{45}$$

Breaking complexity of POs

- **Conditional**:

$$\psi \xleftarrow{c \to R, S} \Phi \quad \equiv \quad \psi \xleftarrow{R} \Phi \cdot \Phi_c \ \wedge \ \psi \xleftarrow{S} \Phi \cdot \Phi_{\neg c} \qquad (46)$$

  which generalizes (6).

NB:

- Close relationship with Hoare logic axioms

  — but note many **equivalences** instead of implications

Exercise 7: Use the PF-calculus to prove the correctness of the rules given above.

□

Checking *FS_DeleteFileDir*:

$$\in_{System \times FFS\_Status} \xleftarrow{FS\_DeleteFileDir\ \ p} \in_{System}$$

$$\equiv \quad \{\ (36)\ \}$$

$$\in_{System} \times id \xleftarrow{c \rightarrow \langle f \cdot \Phi_p, \underline{k} \rangle, \langle id, g \rangle} \in_{System}$$

$$\equiv \quad \{\ \text{conditional } (46)\ \}$$

$$\in_{System} \times id \xleftarrow{\langle f \cdot \Phi_p, \underline{k} \rangle} \in_{System} \cdot \Phi_c$$

$$\wedge$$

$$\in_{System} \times id \xleftarrow{\langle id, g \rangle} \in_{System} \cdot \Phi_{\neg c}$$

Verified File System Project

$\equiv$ { splitting (44,43) }

$$\in_{System} \xleftarrow{f \cdot \Phi_p} \in_{System} \cdot \Phi_c$$

$$\wedge$$

$$id \xleftarrow{\underline{k}} \in_{System} \cdot \Phi_c \cdot \delta\,(f \cdot \Phi_p)$$

$$\wedge$$

$$\in_{System} \xleftarrow{id} \in_{System} \cdot \Phi_{\neg c}$$

$$\wedge$$

$$id \xleftarrow{g} \in_{System} \cdot \Phi_{\neg c}$$

$\equiv$ { (37), (5) }

Case study 3: Verified File System

$$\in_{System} \xleftarrow{f \cdot \Phi_p} \in_{System} \cdot \Phi_c$$

$\equiv$ { trading (38), unfold $\in_{System}$ (35) }

$$(id \times \Phi_{pc}) \cdot \Phi_{ri} \xleftarrow{f \cdot \Phi_p \cdot \Phi_c} (id \times \Phi_{pc}) \cdot \Phi_{ri}$$

$\Leftarrow$ { separating (42) }

$$\Phi_{ri} \xleftarrow{f \cdot \Phi_p \cdot \Phi_c} \Phi_{ri} \quad \wedge \quad id \times \Phi_{pc} \xleftarrow{f \cdot \Phi_p \cdot \Phi_c} id \times \Phi_{pc}$$

$\equiv$ { trading (38) and implication $c \Rightarrow p$ }

$$\Phi_{ri} \xleftarrow{f} \Phi_{ri} \cdot \Phi_c \quad \wedge$$

$$id \times \Phi_{pc} \xleftarrow{f} (id \times \Phi_{pc}) \cdot \Phi_c$$

Case study 3: Verified File System

- So much for PO calculation "in-the-large".
- Going "in-the-small" means spelling out invariants, functions and pre-conditions and reason as in the previous case studies
- Let us pick the first PO, $\Phi_{ri} \xleftarrow{f} \Phi_{ri} \cdot \Phi_c$, for example.
- As earlier on, we go pointwise and try to rewrite $ri(f(M, N))$ — M keeps open file descriptors, N the file contents — into $ri(M, N)$ + a weakest precondition; then we compare the outcome with what the designer wrote (Φ_c).

Case study 3: Verified File System

Clearly, from slide 25 we infer

$$ri(M, N) \quad \triangleq \quad \rho\,(path \cdot M) \subseteq \delta\, N$$

cf. diagram

$$\begin{tikzcd}
FileHandler \arrow[r, harpoon, "M"] & OpenFileDescriptor \arrow[dl, "path"] \\
Path \arrow[r, harpoon, "N"'] & File
\end{tikzcd}$$

which is a referential integrity constraint relating paths in open-file descriptors and paths in the file store N. PF calculation will lead to

$$ri(M, N) \;\; \triangleq \;\; path \cdot M \subseteq N^\circ \cdot \top \tag{47}$$

thanks to (64) etc.

Case study 3: Verified File System

We calculate:

$$
\begin{array}{ll}
 & ri(f(M,N)) \\
= & \qquad \{\ (36)\ \} \\
 & ri(FS\_DeleteFileDir\_System\ p\ (M,N)) \\
= & \qquad \{\ FS\_DeleteFileDir\_Tar\ s\ t \triangleq\ tar \setminus s \text{ etc }\ \} \\
 & ri(M, N \cdot \neg\rho\,\underline{p})
\end{array}
$$

We can generalize from single path p to a set S of paths:

$$
\begin{array}{ll}
 & ri(M, N \cdot \Phi_{\neg S}) \\
\equiv & \qquad \{\ (47)\ \} \\
 & path \cdot M \subseteq (N \cdot \Phi_{\neg S})^{\circ} \cdot \top \\
\equiv & \qquad \{\ \text{converses } (50,51,\ 54)\ \}
\end{array}
$$

Case study 3: Verified File System

$$
\begin{array}{ll}
 & path \cdot M \subseteq \Phi_{\neg S} \cdot N^{\circ} \cdot \top \\
\equiv & \qquad \{ \text{ (66), coreflexives (55), } (\cdot\top) \text{ distribution } \} \\
 & path \cdot M \subseteq \Phi_{\neg S} \cdot \top \cap N^{\circ} \cdot \top \\
\equiv & \qquad \{ \text{ } \cap\text{-universal (52) } \} \\
 & path \cdot M \subseteq \Phi_{\neg S} \cdot \top \ \wedge \ path \cdot M \subseteq N^{\circ} \cdot \top \\
\equiv & \qquad \{ \text{ “al-djabr” ; (47) } \} \\
 & \underbrace{M \subseteq path^{\circ} \cdot \Phi_{\neg S} \cdot \top}_{\text{wp}} \qquad \wedge \, ri(M, N) \\
\equiv & \qquad \{ \text{ going pointwise } \} \\
 & \langle \forall \ b \ : \ b \in rng \ M : \ path \ b \notin S \rangle \ \wedge \ ri(M, N)
\end{array}
$$

Summary

- Thus we've checked (part) of the pre-condition of *FS_DeleteFileDir_System*, recall slide 25
- The other checks are performed in a similar way.
- Two levels of PO calculation: **in-the-large** (PO level) and **in-the-small** (where PF-notation describes data).
- PO-level useful in preparing POs for a theorem prover, recall diagram of slide 3.

Background

"Napkin" rule:

$$b(f^\circ \cdot R \cdot g)a \quad \equiv \quad (f\ b)R(g\ a) \tag{48}$$

Converses:

$$\begin{aligned}
(R \cup S)^\circ &= R^\circ \cup S^\circ && (49)\\
(R \cdot S)^\circ &= S^\circ \cdot R^\circ && (50)\\
(R^\circ)^\circ &= R && (51)
\end{aligned}$$

Background

Meet and join:

$$X \subseteq R \cap S \equiv X \subseteq R \wedge X \subseteq S \tag{52}$$

$$R \cup S \subseteq X \equiv R \subseteq X \wedge S \subseteq X \tag{53}$$

Coreflexives are **symmetric** and **transitive**:

$$\Phi^\circ = \Phi = \Phi \cdot \Phi \tag{54}$$

Meet of two coreflexives is composition:

$$\Phi \cap \Psi = \Phi \cdot \Psi \tag{55}$$

Background

Equality on relations $B \xleftarrow{R,S} A$:

$$R = S \quad \equiv \quad R \subseteq S \wedge S \subseteq R \tag{56}$$

Alternative to (56) — **indirect equality** rules:

$$R = S \quad \equiv \quad \langle \forall\, X \;::\; (X \subseteq R \equiv X \subseteq S)\rangle \tag{57}$$

$$\equiv \quad \langle \forall\, X \;::\; (R \subseteq X \equiv S \subseteq X)\rangle \tag{58}$$

Shunting rules:

$$f \cdot R \subseteq S \quad \equiv \quad R \subseteq f^{\circ} \cdot S \tag{59}$$

$$R \cdot f^{\circ} \subseteq S \quad \equiv \quad R \subseteq S \cdot f \tag{60}$$

Therefore

$$f \cdot (R \cup S) \quad = \quad f \cdot R \cup f \cdot S \tag{61}$$

Background

Kernel, image (the same for respectively δ, ρ):

$$\ker(R^\circ) = \operatorname{img} R \tag{62}$$

$$\operatorname{img}(R^\circ) = \ker R \tag{63}$$

Range:

$$\rho R \subseteq \Phi \equiv R \subseteq \Phi \cdot \top \tag{64}$$

Domain/range elimination:

$$\top \cdot \delta R = \top \cdot R \tag{65}$$

$$\rho R \cdot \top = R \cdot \top \tag{66}$$

Background

Weakest (liberal) pre-condition is the upper adjoint of the following "al-djabr" rule [1] which combines two already seen — range (64) and left division (13):

$$\rho\,(R \cdot \Phi) \subseteq \Psi \quad \equiv \quad \Phi \subseteq R \setminus \Psi \tag{67}$$

The pointwise version $wlp\ R\ \psi$ of $R \setminus \Psi$ is:

$$wlp\ R\ \psi \quad \triangleq \quad \langle \bigvee\ \phi\ ::\ \langle \forall\ b, a\ :\ b\ R\ a :\ \phi\ a \Rightarrow \psi\ b \rangle\rangle$$

In the slide which follows we show that, if equivalence (23) holds then *pre* is the weakest precondition for *inv* to be maintained. The calculation proceeds by **indirect equality** (57) over coreflexive Φ:

Weakest pre-conditions

$$\Phi \subseteq \Phi_{pre} \cdot \Phi_{inv}$$

$\equiv$ { rep. equal by equals (24) }

$$\Phi \subseteq \top \cdot \Phi_{inv} \cdot f$$

$\equiv$ { "al-djabr" rule (59) ; converses }

$$f \cdot \Phi \subseteq \Phi_{inv} \cdot \top$$

$\equiv$ { range (64) }

$$\rho\,(f \cdot \Phi) \subseteq \Phi_{inv}$$

$\equiv$ { weakest pre-condition (67) }

$$\Phi \subseteq f \mathbin{\backslash\!\!\!\bullet} \Phi_{inv}$$

$::$ { indirection (57) }

$$\Phi_{pre} \cdot \Phi_{inv} = f \mathbin{\backslash\!\!\!\bullet} \Phi_{inv}$$

R.C. Backhouse.
Fixed point calculus, 2000.
Summer School and Workshop on Algebraic and Coalgebraic Methods in the Mathematics of Program Construction, Lincoln College, Oxford, UK 10th to 14th April 2000.

Rajeev Joshi and Gerard J. Holzmann.
A mini challenge: build a verifiable filesystem.
Formal Asp. Comput., 19(2):269–272, 2007.